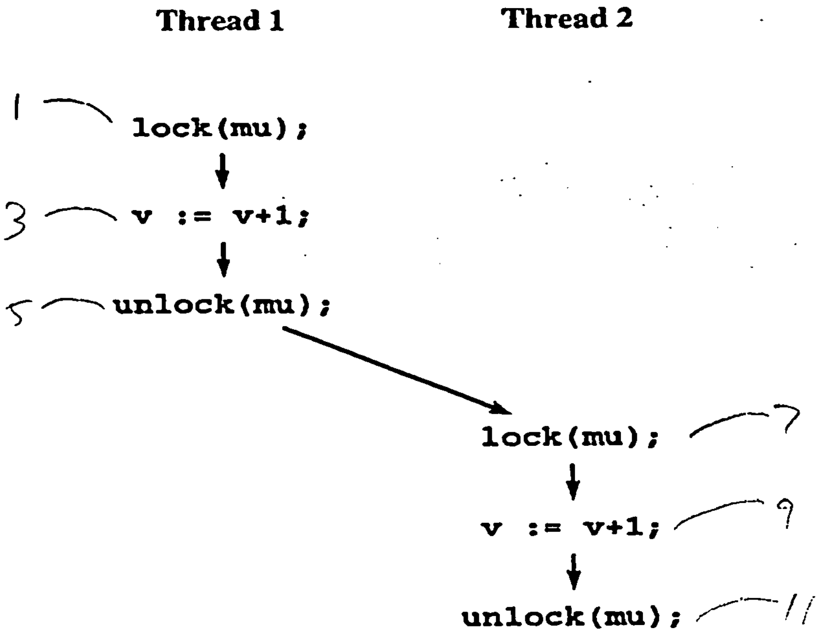


FIG. 1

```
      Program              locks_held      C(v)
                              {}         {mu1,mu2}
101   lock(mu1);
                            {mu1}
103   v := v+1;
                                           {mu1}
105   unlock(mu1);
                              {}
107   lock(mu2);
                            {mu2}
109   v := v+1;
                                            {}
111   unlock(mu2);
                              {}
```

FIG. 2

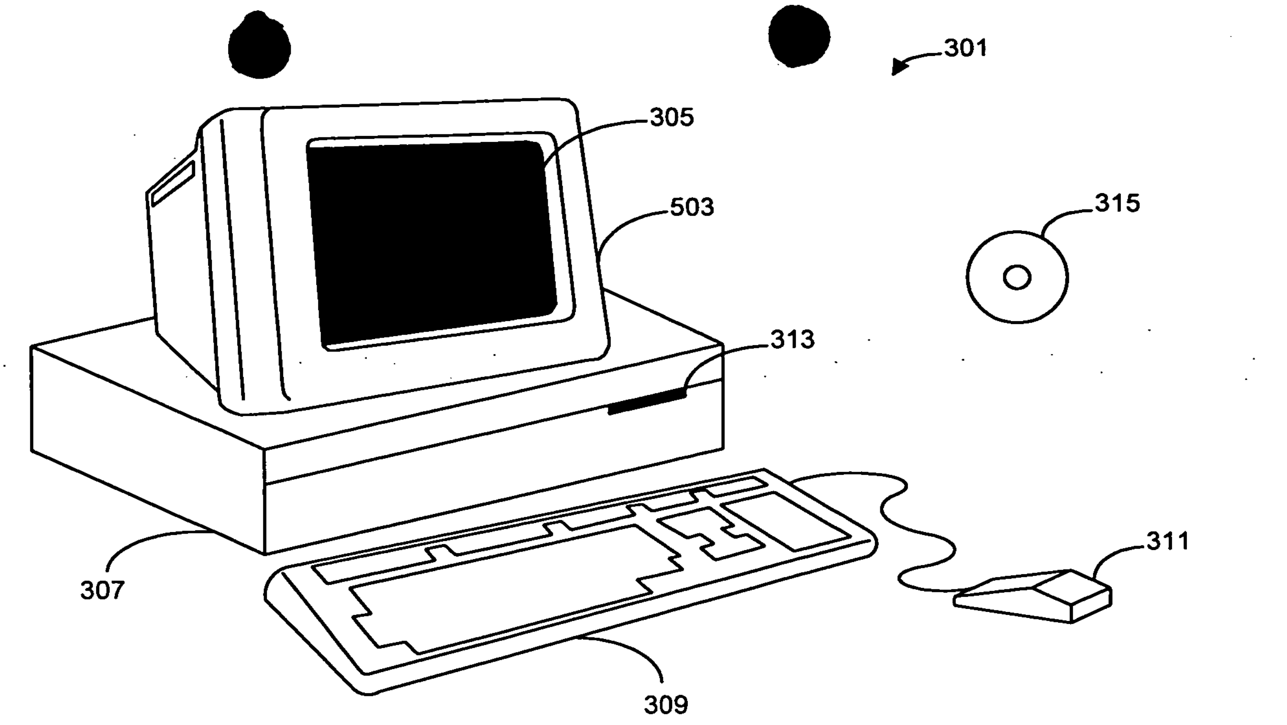

301
305
503
315
313
311
307
309

FIG. 3

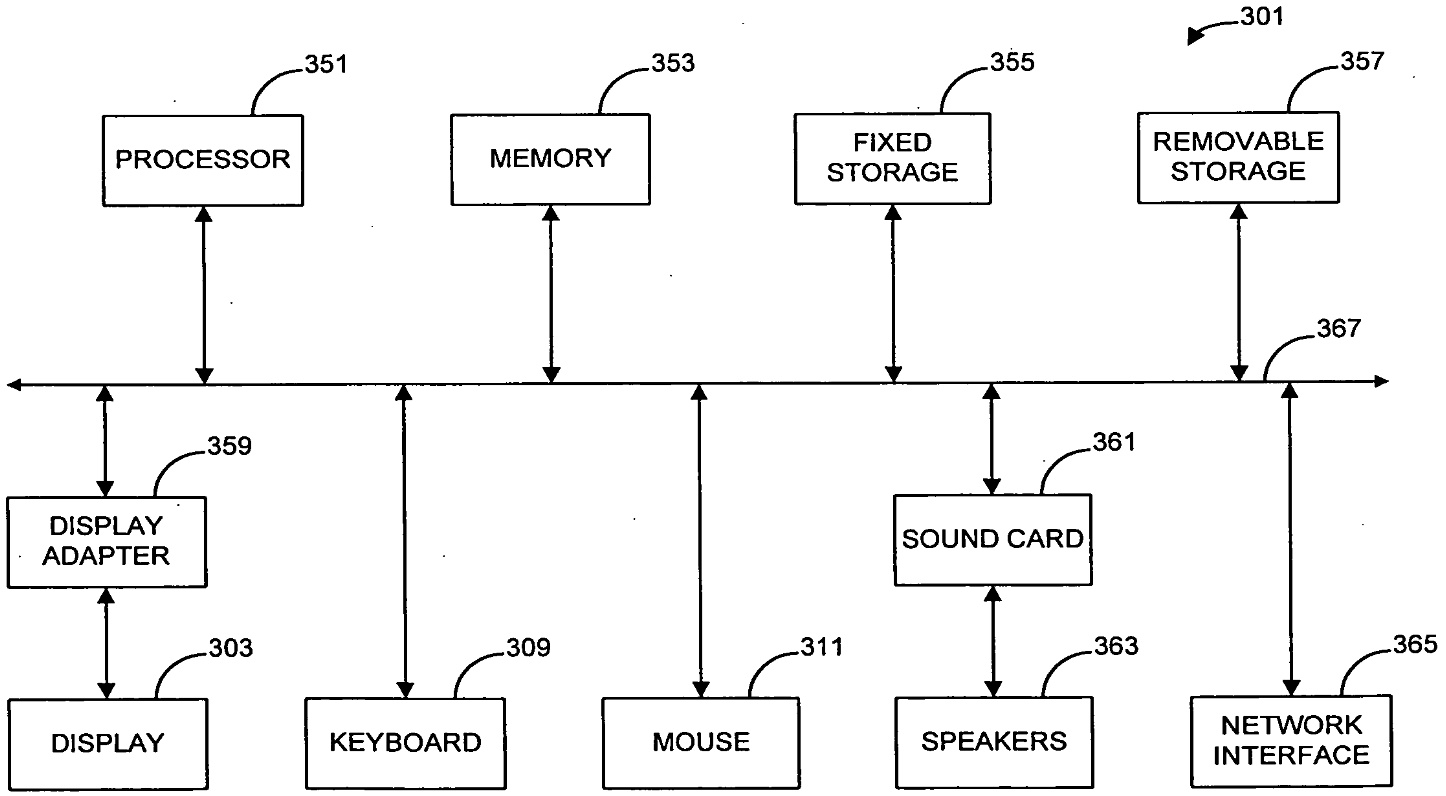

301
351
PROCESSOR
353
MEMORY
355
FIXED STORAGE
357
REMOVABLE STORAGE
367
359
DISPLAY ADAPTER
361
SOUND CARD
303
DISPLAY
309
KEYBOARD
311
MOUSE
363
SPEAKERS
365
NETWORK INTERFACE

FIG. 4

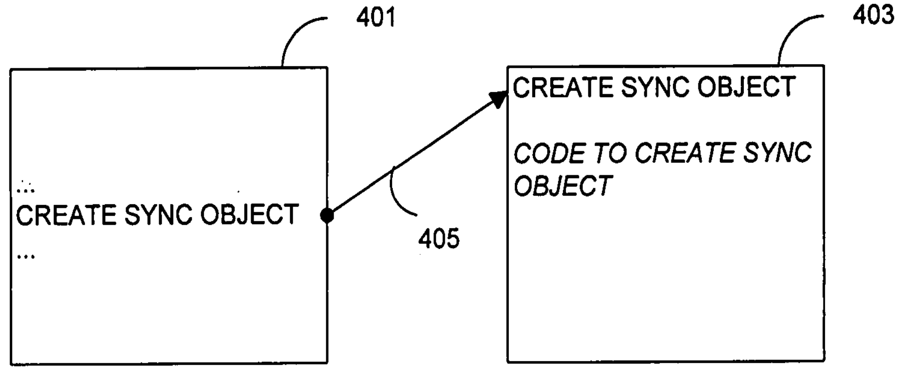

401
...
CREATE SYNC OBJECT
...
405
403
CREATE SYNC OBJECT
CODE TO CREATE SYNC OBJECT

FIG. 5A

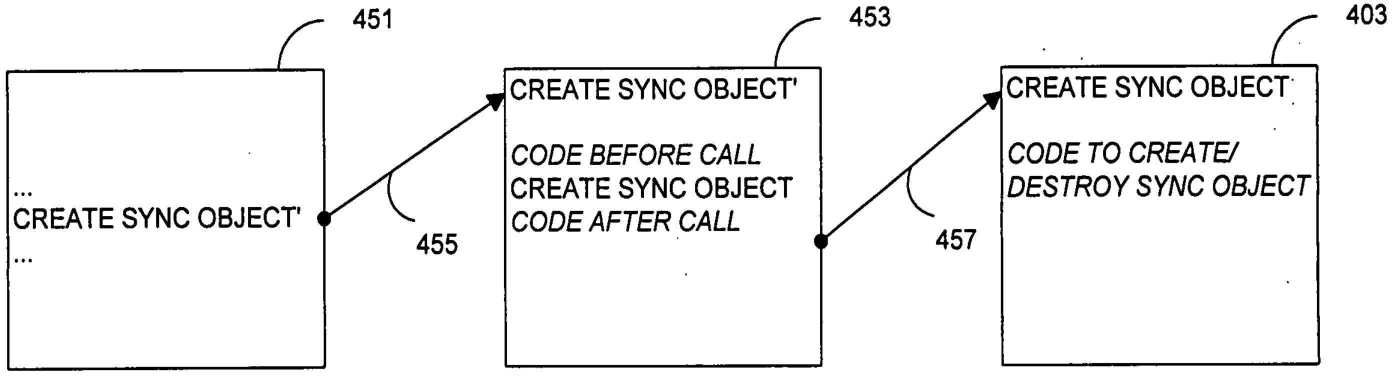

451
...
CREATE SYNC OBJECT'
...
455
453
CREATE SYNC OBJECT'
CODE BEFORE CALL
CREATE SYNC OBJECT
CODE AFTER CALL
457
403
CREATE SYNC OBJECT
CODE TO CREATE/ DESTROY SYNC OBJECT

FIG. 5B

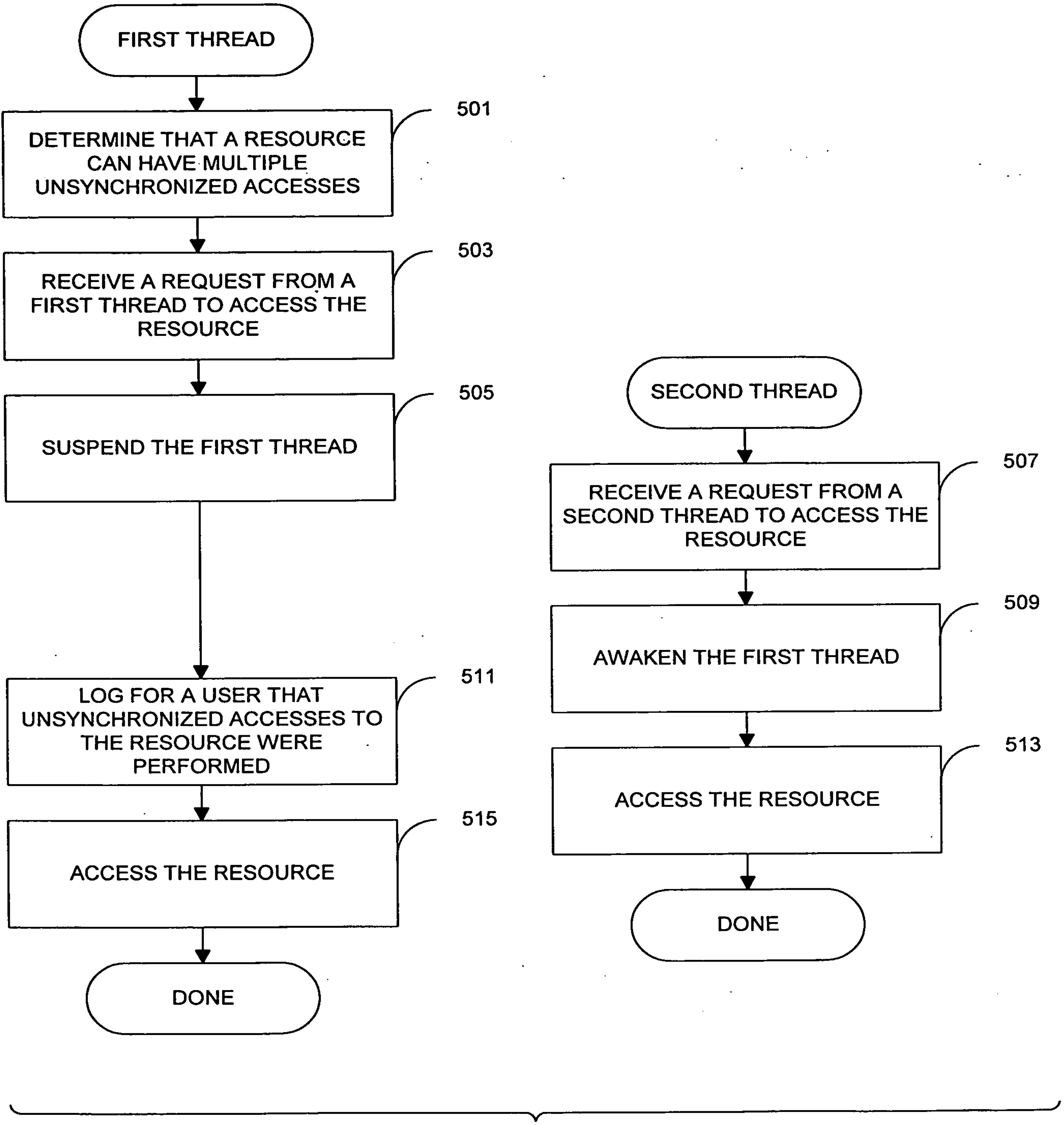

FIRST THREAD
DETERMINE THAT A RESOURCE CAN HAVE MULTIPLE UNSYNCHRONIZED ACCESSES
501
RECEIVE A REQUEST FROM A FIRST THREAD TO ACCESS THE RESOURCE
503
SUSPEND THE FIRST THREAD
505
LOG FOR A USER THAT UNSYNCHRONIZED ACCESSES TO THE RESOURCE WERE PERFORMED
511
ACCESS THE RESOURCE
515
DONE
SECOND THREAD
RECEIVE A REQUEST FROM A SECOND THREAD TO ACCESS THE RESOURCE
507
AWAKEN THE FIRST THREAD
509
ACCESS THE RESOURCE
513
DONE

FIG. 6

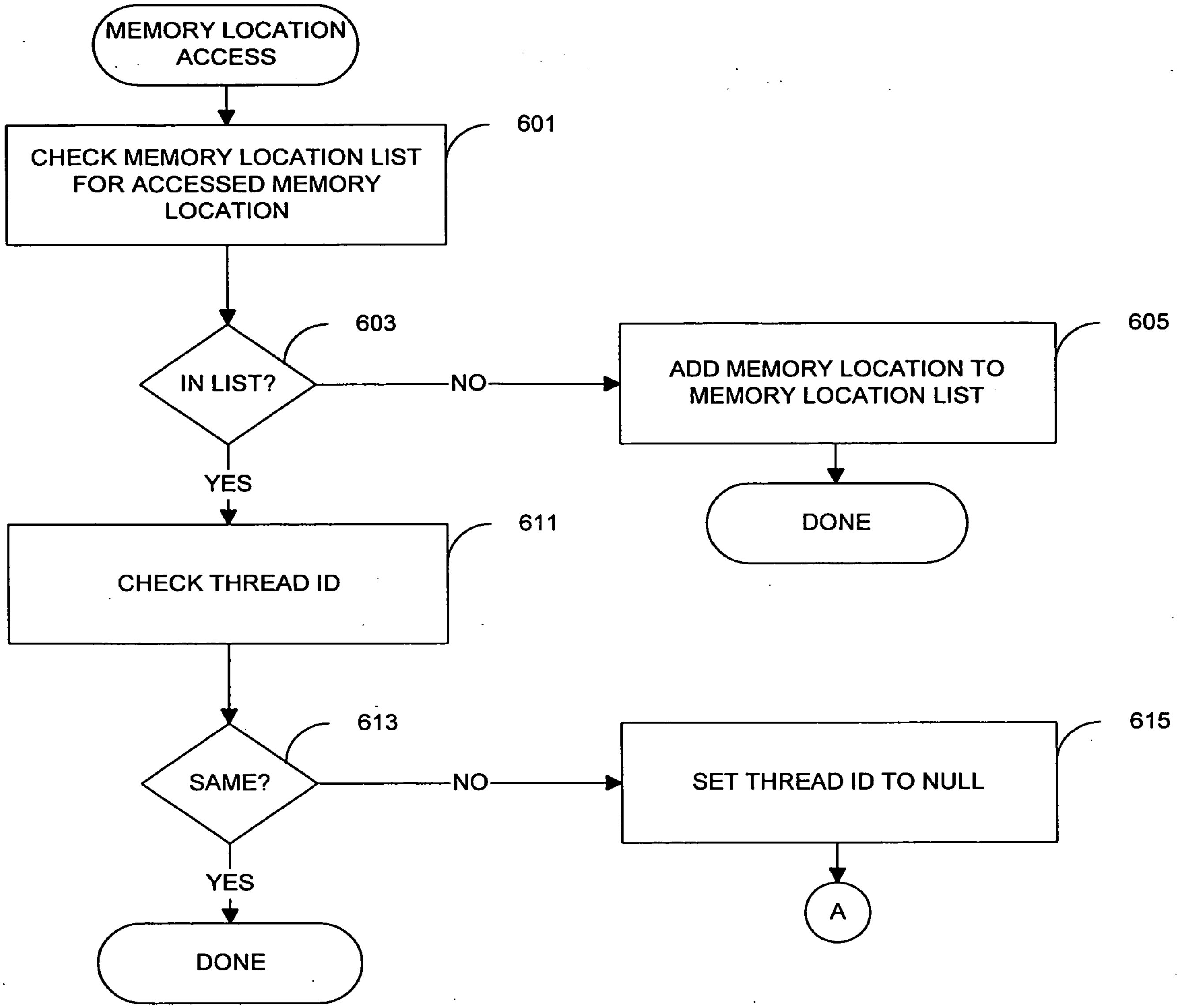

MEMORY LOCATION ACCESS
CHECK MEMORY LOCATION LIST FOR ACCESSED MEMORY LOCATION
601
IN LIST?
603
NO
ADD MEMORY LOCATION TO MEMORY LOCATION LIST
605
DONE
YES
CHECK THREAD ID
611
SAME?
613
NO
SET THREAD ID TO NULL
615
A
YES
DONE

FIG. 7A

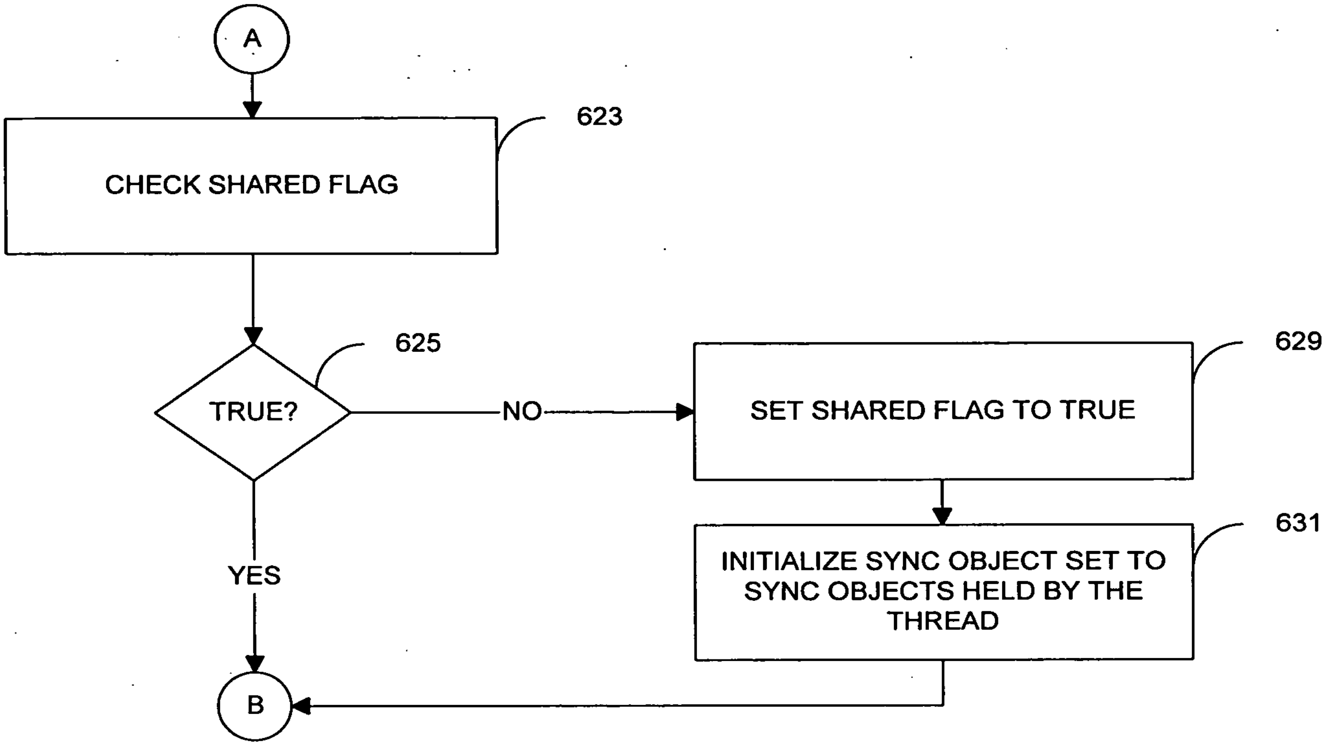

A
623
CHECK SHARED FLAG
625
TRUE?
NO
629
SET SHARED FLAG TO TRUE
631
INITIALIZE SYNC OBJECT SET TO SYNC OBJECTS HELD BY THE THREAD
YES
B

FIG. 7B

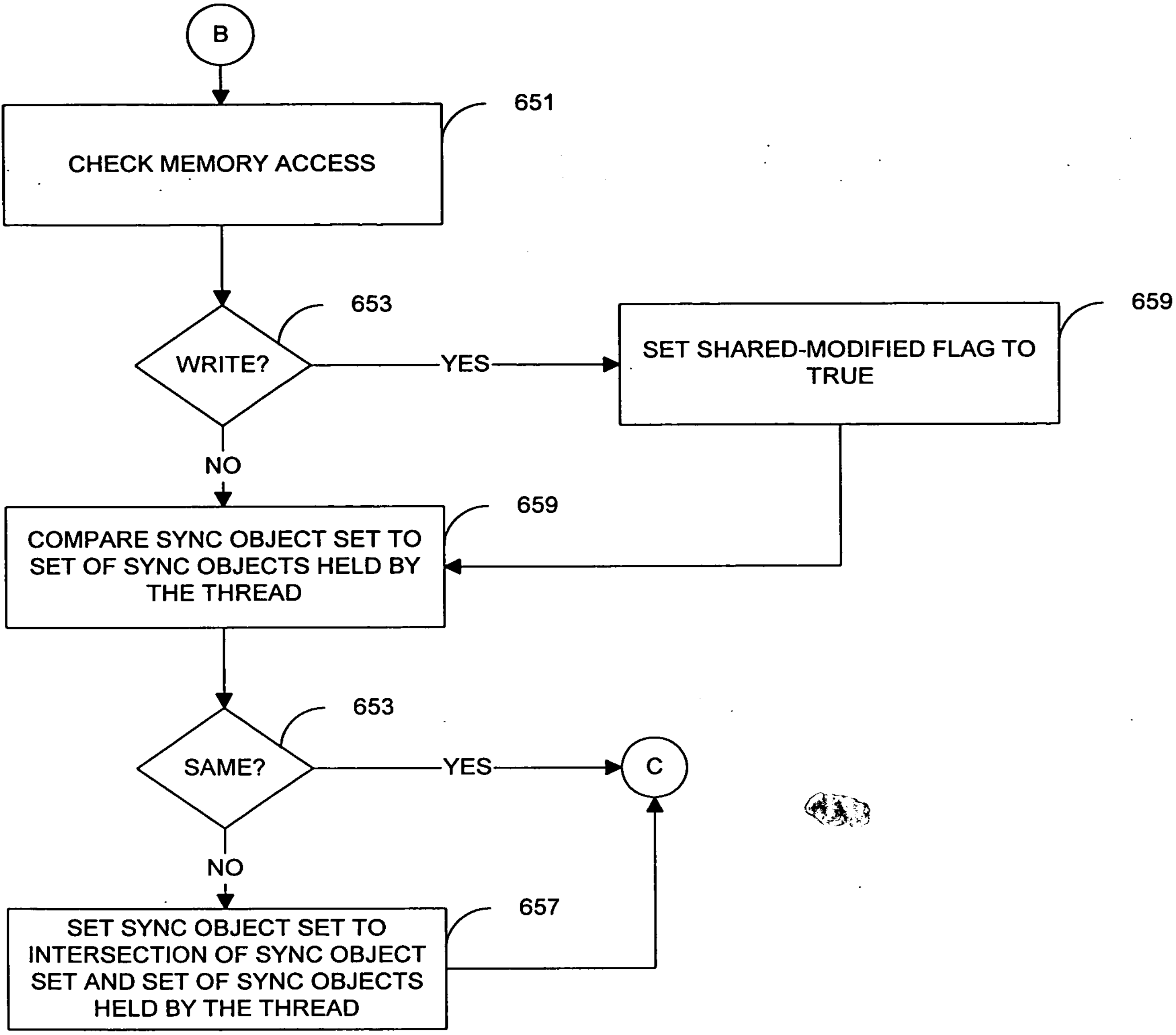

B
CHECK MEMORY ACCESS
651
WRITE?
653
YES
SET SHARED-MODIFIED FLAG TO TRUE
659
NO
COMPARE SYNC OBJECT SET TO SET OF SYNC OBJECTS HELD BY THE THREAD
659
SAME?
653
YES
C
NO
SET SYNC OBJECT SET TO INTERSECTION OF SYNC OBJECT SET AND SET OF SYNC OBJECTS HELD BY THE THREAD
657

FIG. 7C

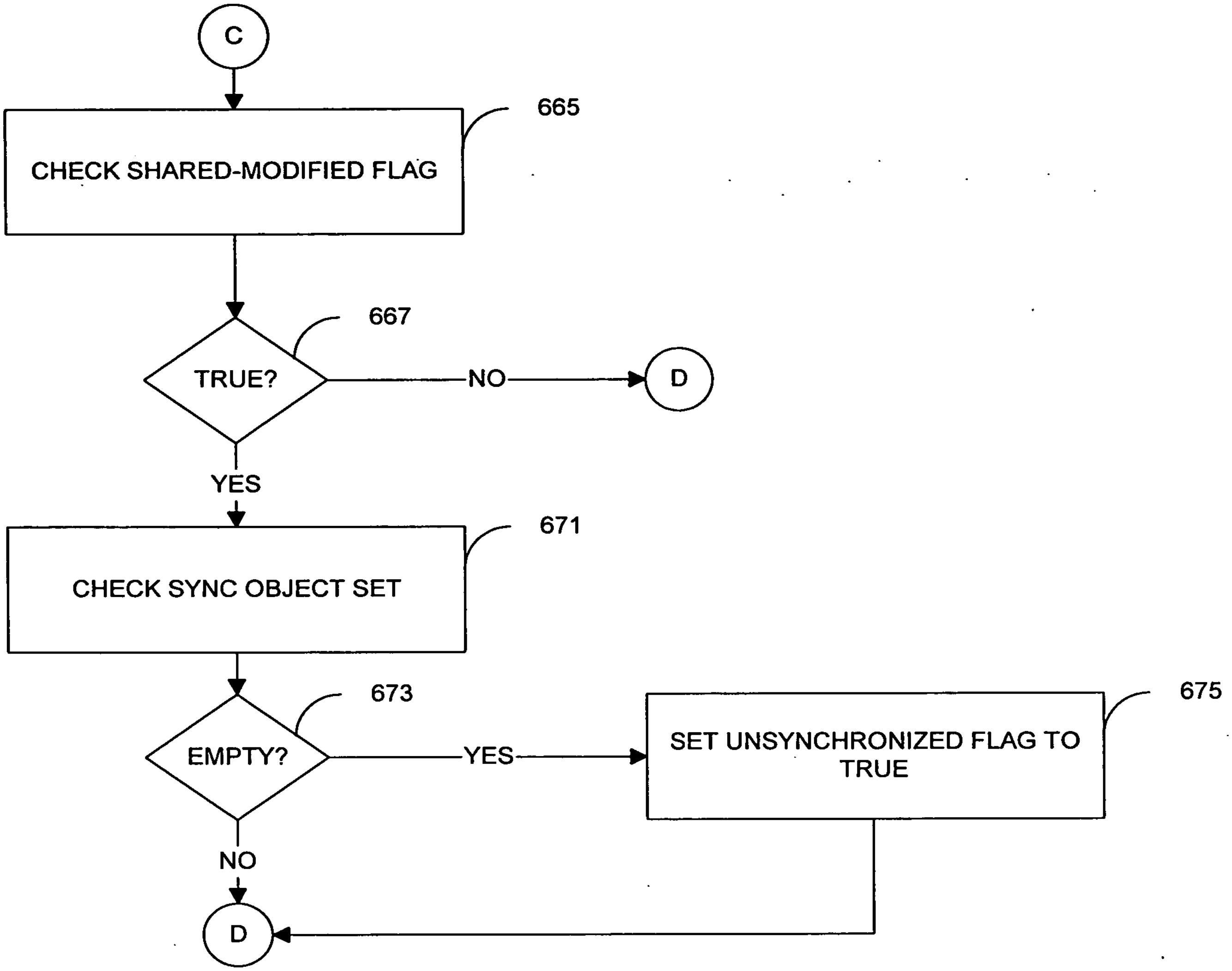

C
CHECK SHARED-MODIFIED FLAG
665
TRUE?
667
NO
D
YES
CHECK SYNC OBJECT SET
671
EMPTY?
673
YES
SET UNSYNCHRONIZED FLAG TO TRUE
675
NO
D

FIG. 7D

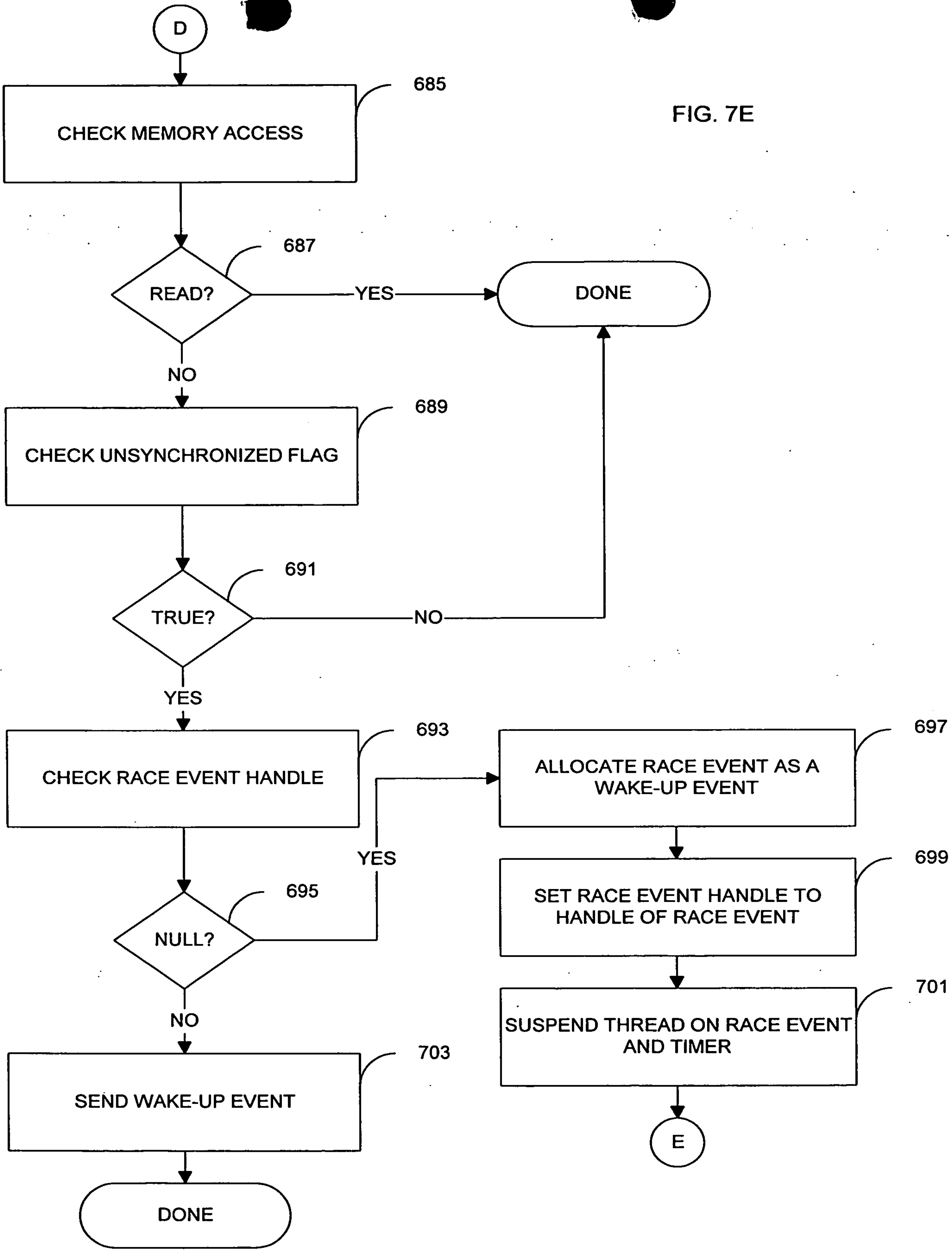

D
685
CHECK MEMORY ACCESS
687
READ?
YES
DONE
NO
689
CHECK UNSYNCHRONIZED FLAG
691
TRUE?
NO
YES
693
CHECK RACE EVENT HANDLE
697
ALLOCATE RACE EVENT AS A WAKE-UP EVENT
YES
695
NULL?
699
SET RACE EVENT HANDLE TO HANDLE OF RACE EVENT
701
SUSPEND THREAD ON RACE EVENT AND TIMER
NO
703
SEND WAKE-UP EVENT
E
DONE

FIG. 7E

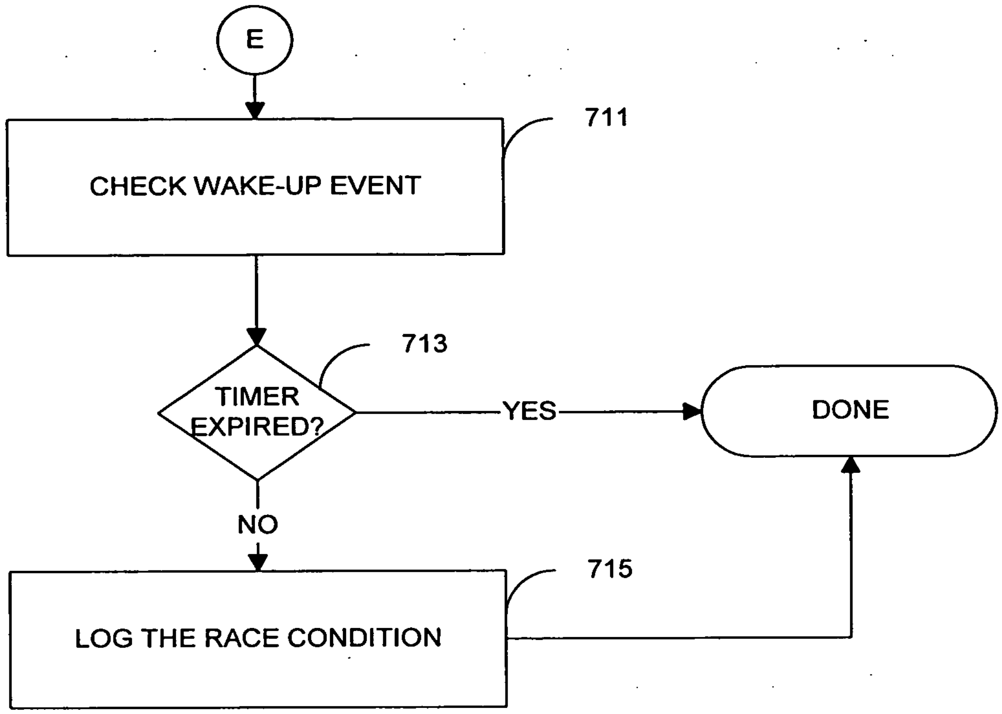

E
CHECK WAKE-UP EVENT
711
TIMER EXPIRED?
713
YES
DONE
NO
LOG THE RACE CONDITION
715

FIG. 7F

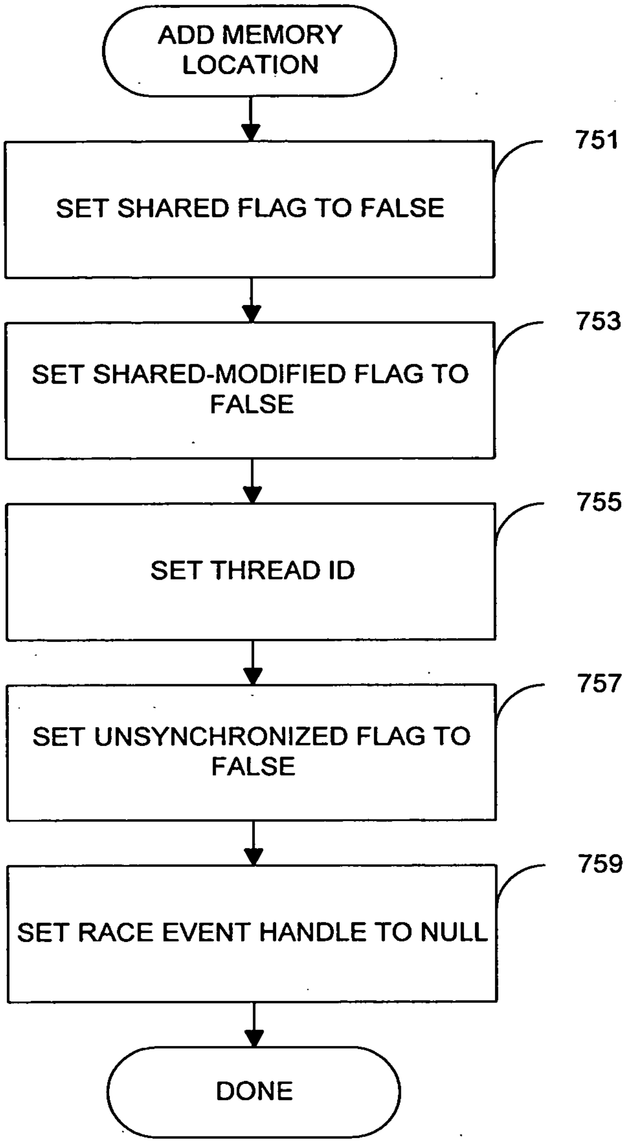
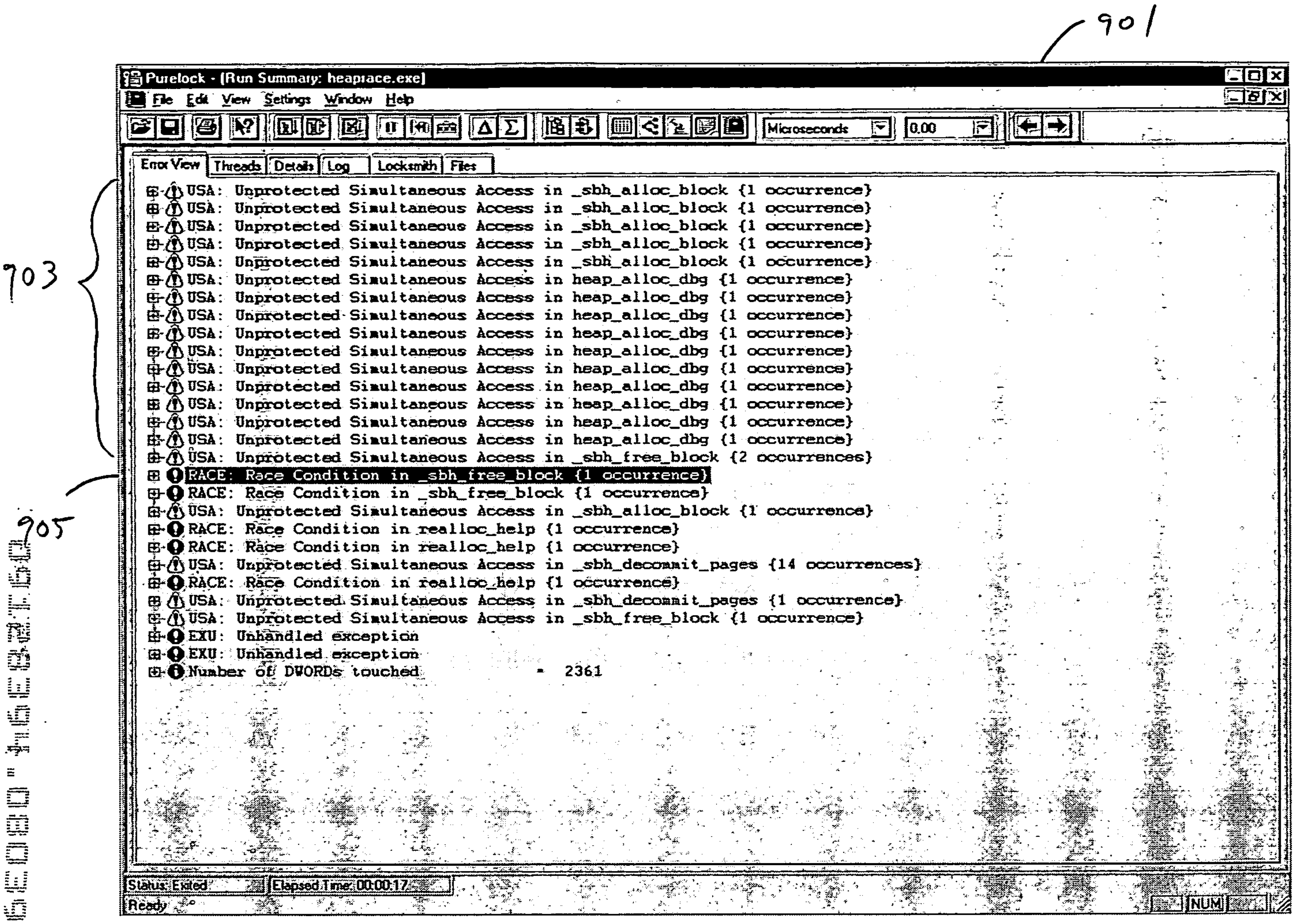
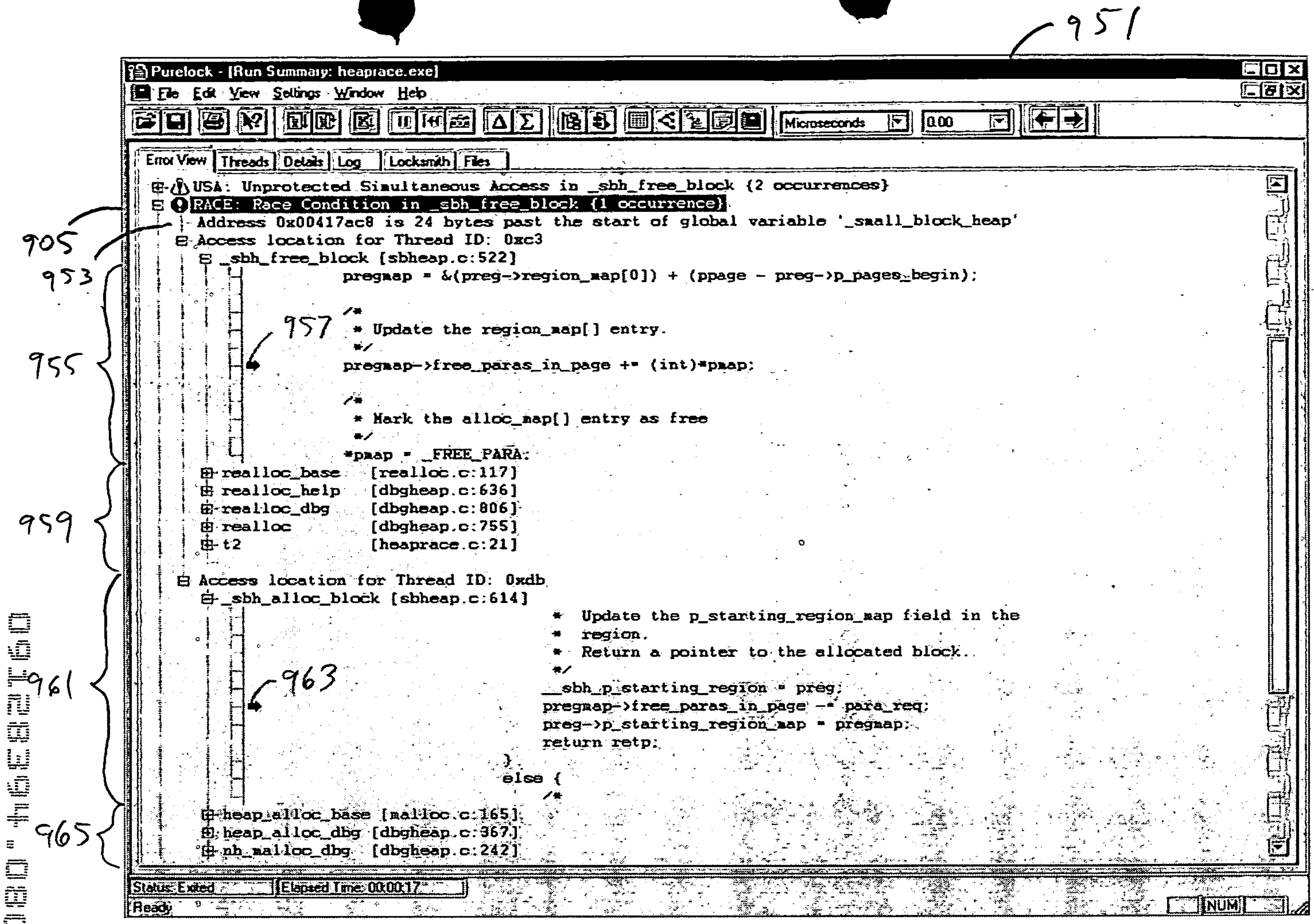

ADD MEMORY LOCATION
751
SET SHARED FLAG TO FALSE
753
SET SHARED-MODIFIED FLAG TO FALSE
755
SET THREAD ID
757
SET UNSYNCHRONIZED FLAG TO FALSE
759
SET RACE EVENT HANDLE TO NULL
DONE

FIG. 8



```
⊞ USA: Unprotected Simultaneous Access in _sbh_alloc_block {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_alloc_block {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_alloc_block {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_alloc_block {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_alloc_block {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in heap_alloc_dbg {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_free_block {2 occurrences}
⊞ RACE: Race Condition in _sbh_free_block {1 occurrence}
⊞ RACE: Race Condition in _sbh_free_block {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_alloc_block {1 occurrence}
⊞ RACE: Race Condition in realloc_help {1 occurrence}
⊞ RACE: Race Condition in realloc_help {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_decommit_pages {14 occurrences}
⊞ RACE: Race Condition in realloc_help {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_decommit_pages {1 occurrence}
⊞ USA: Unprotected Simultaneous Access in _sbh_free_block {1 occurrence}
⊞ EXU: Unhandled exception
⊞ EXU: Unhandled exception
⊞ Number of DWORDs touched      =   2361
```

FIG. 9



```
⊞ USA: Unprotected Simultaneous Access in _sbh_free_block (2 occurrences)
⊟ RACE: Race Condition in _sbh_free_block (1 occurrence)
   Address 0x00417ac8 is 24 bytes past the start of global variable '_small_block_heap'
  ⊟ Access location for Thread ID: 0xc3
    ⊟ _sbh_free_block [sbheap.c:522]
                  pregmap = &(preg->region_map[0]) + (ppage - preg->p_pages_begin);

                  /*
                   * Update the region_map[] entry.
                   */
                  pregmap->free_paras_in_page += (int)*pmap;

                  /*
                   * Mark the alloc_map[] entry as free
                   */
                  *pmap = _FREE_PARA;
      ⊞ realloc_base   [realloc.c:117]
      ⊞ realloc_help   [dbgheap.c:636]
      ⊞ realloc_dbg    [dbgheap.c:806]
      ⊞ realloc        [dbgheap.c:755]
      ⊞ t2             [heaprace.c:21]
  ⊟ Access location for Thread ID: 0xdb
    ⊟ _sbh_alloc_block [sbheap.c:614]
                              *  Update the p_starting_region_map field in the
                              *  region.
                              *  Return a pointer to the allocated block.
                              */
                            __sbh_p_starting_region = preg;
                            pregmap->free_paras_in_page -= para_req;
                            preg->p_starting_region_map = pregmap;
                            return retp;
                        }
                        else {
                            /*
      ⊞ heap_alloc_base [malloc.c:165]
      ⊞ heap_alloc_dbg [dbgheap.c:367]
      ⊞ nh_malloc_dbg  [dbgheap.c:242]
```

FIG. 10